**1. CSRF คืออะไร**

• ป้องกันการโจมตีที่เกิดจาก browser อื่น (เช่น เว็บปลอม) ส่ง request มายัง API ที่ user login อยู่

• ใช้ได้ผลกับ request ที่ browser ส่งโดยอัตโนมัติ (เช่น cookie/session)

**2. ควรใช้ CSRF กับ endpoint ไหน**

• ควรใช้:

o ทุก endpoint ที่ "เปลี่ยนแปลงข้อมูล" (POST/PUT/PATCH/DELETE)

o เช่น /api/patient/create, /api/patient/update, /api/appointment/delete

• ไม่ควรใช้:

o endpoint ที่ "ไม่ต้องการ session/cookie" เช่น login, guest login, public API

**3. login ไม่ควรใช้ CSRF**

• เพราะ user ยังไม่มี session/cookie

• ถ้าใช้ CSRF กับ login จะทำให้ flow ซับซ้อนโดยไม่จำเป็น

• ส่วนใหญ่ระบบ auth จะใช้แค่ username/password + rate limit + brute force protection

4. ตัวอย่าง flow ที่เหมาะสม

1. POST /auth/login

o ไม่ต้องใช้ CSRF

o รับ username/password → ออก token/cookie

2. GET /auth/csrf-token

o หลัง login แล้ว (browser มี cookie แล้ว)

o ขอ CSRF token สำหรับใช้กับ request ที่เปลี่ยนแปลงข้อมูล

3. POST /patient/create

o ต้องแนบ CSRF token

o Backend ตรวจสอบ CSRF ก่อนดำเนินการ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

สรุปแนวทางที่เหมาะสม

• Login: ไม่ต้องใช้ CSRF

• API ที่แก้ไขข้อมูล: ใช้ CSRF

• API ที่อ่านข้อมูล (GET): ไม่ต้องใช้ CSRF

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

อ้างอิง:

• OWASP CSRF Prevention Cheat Sheet

• NestJS CSRF Example

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ตอบโจทย์:

• ระบบ login ไม่ต้องใช้ CSRF

• ใช้ CSRF กับ API ที่เปลี่ยนแปลงข้อมูลหลัง login (เช่น table คนไข้)

• ป้องกันการโจมตีจาก browser อื่นที่พยายามเปลี่ยนแปลงข้อมูลของ user



